IDENTIFICACIÓN DE SISTEMAS NO LINEALES CON UN MODELO HAMMERSTEIN-WIENER APLICADO A UN GENERADOR EÓLICO

MARIO ALEJANDRO USECHE ARTEAGA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PROGRAMA DE INGENIERÍA ELÉCTRICA
PEREIRA
2013

IDENTIFICACIÓN DE SISTEMAS NO LINEALES CON UN MODELO HAMMERSTAIN-WIENNER APLICADO A UN GENERADOR EÓLICO

MARIO ALEJANDRO USECHE ARTEAGA

Trabajo de grado para optar al título de Ingeniero Electricista

Director
Ph.D. EDUARDO GIRALDO SUÁREZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PROGRAMA DE INGENIERÍA ELÉCTRICA
PEREIRA
2013

Índice general

Índice de figuras

Resumen

Se presentan metodologías para la identificación de sistemas multivariables lineales y no lineales aplicados a un sistema de generación de energía eólica a través de la teoría de identificación por subespacio de estados para sistemas lineales, un modelo Hammerstein-Wienner y las máquinas de soporte vectorial.

Abstract

we present a methodology for the identification of linear and non-linear multivariable systems aplied to the wind power generation Systems, using the subspace identification for linear systems, a Hammerstein-Wienner model and the soport vector machine.

Introducción

La identificación de sistemas permite construir modelos que representan el comportamiento real de sistemas, a partir de la observación de datos de las señales de entrada y salida del sistema, se busca predecir y controlar el comportamiento del sistema a través del tiempo [2].

La linealización de los sistemas permite modelar los sistemas no lineales por medio de modelos lineales alrededor de un rango de operación, reduciendo la complejidad del sistema de identificación y, generalmente, presentando resultados aceptables. Sin embargo en algunos casos, estos tipos de modelos introducen muchas aproximaciones que no reflejan adecuadamente el comportamiento real del sistema no lineal y es necesario utilizar modelos de mayor complejidad [2].

La energía eólica es una energía renovable, amigable con el ambiente; por ello ha sido desarrollada con mayor rapidez por el incremento en el tamaño, la capacidad de las turbinas y la cantidad de turbinas instaladas en los parques eólicos. Sin embargo el sistema de energía eólica es un sistema no lineal que depende de la velocidad del viento y de las características atmosféricas del ambiente [7]. Por lo tanto, es necesario identificar e implementar el control del generador para minimizar los efectos de la variación en las características del viento, la atmósfera y las perturbaciones, mejorando la calidad de la energía generada por el sistema eólico.

Capítulo 1

Identificación por subespacios de estados para sistemas lineales

El algoritmo de identificación por subespacios identifica modelos de la forma:

$$\begin{aligned} x_{t+1} &= A.x_t + B.u_t \\ y_t &= C.x_t + D.u_t \end{aligned} \tag{1.1}$$

1.1. Herramientas geométricas

Los algoritmos de identificación por subespacios esta frecuentemente basados en conceptos geométricos, por ello en la siguiente sección se presentaran los conceptos geometricos más relevantes utilizados a lo largo del documento.

1.1.1. Proyección ortogonal

$\prod_B$ denota la proyección del espacio de filas de una matriz dentro del espacio de filas de la matriz $B \in \mathbb{R}^{qxj}$, como se muestra en la ecuación (1.2).

$$\prod_B = B^T * (B * B^T)^\dagger * B \tag{1.2}$$

Donde $(*)^\dagger$ denota la pseudo-inversa Moore-Penrosede la matriz *. $A/\mathbf{B}$ es la proyección del espacio de filas de la matriz A sobre el espacio de filas de la matriz B y se puede calcular como lo muestra la (1.4):

$$A/\mathbf{B} = A\prod_B \tag{1.3}$$

$$= B^T * (B * B^T)^\dagger * B \tag{1.4}$$

El operador de proyección puede ser interpretado en el ambiente del espacio j-dimensional como se indica en la figura 1.1.

$$A/\boldsymbol{B}^\perp = A\prod_{B^\perp} \tag{1.5}$$

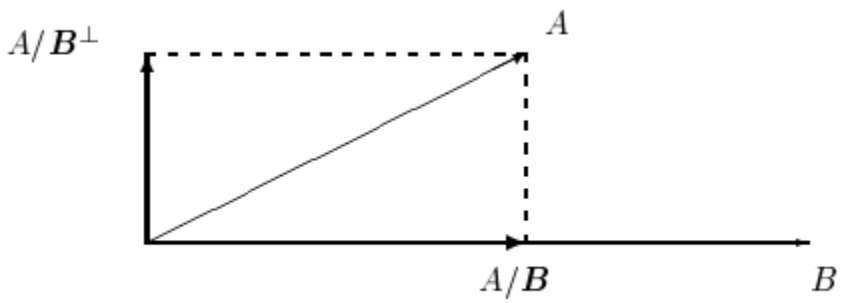

Figura 1.1: Proyección ortogonal

Donde $A\prod_{B^{\perp}}$ es el operador geometrico que denota la proyección del espacio de filas de una matriz dentro del complemento ortogonal del espacio de filas de la matriz B:

$$\prod_{B^{\perp}} = I_j - \prod_{B} \tag{1.6}$$

La combinación de las proyecciones $\prod_B$ y $\prod_{B^{\perp}}$ permiten descomponer la matriz A dentro de dos matrices con sus espacios de filas ortogonales:

$$A = A\prod_{B} + A\prod_{B^{\perp}} \tag{1.7}$$

1.1.2. Proyección oblicua

La proyección oblicua permite descomponer A como una combinacion de dos matrices no ortogonales y el complemento ortogonal de B y C, esto se puede ilustrar en la figura 1.2

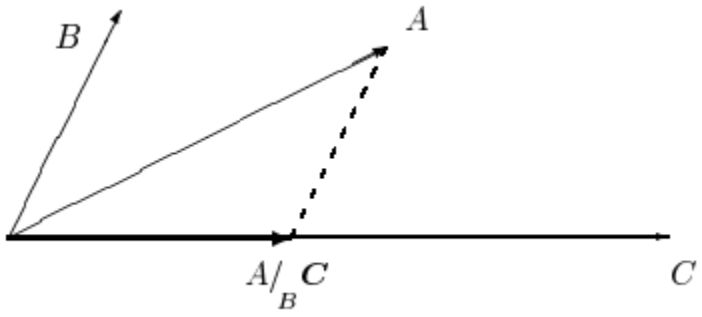

Figura 1.2: Proyección oblicua

La proyección oblicua del espacio de filas de $A \in \mathbb{R}^{pxj}$ a lo largo del espacio de filas de $B \in \mathbb{R}^{qxj}$

sobre el espacio de filas de $C \in \mathbb{R}^{rxj}$ es definida en la ecuación (1.9):

$$A/_B\boldsymbol{C} = \left[A/\boldsymbol{B^T}\right].\left[C/\boldsymbol{B^T}\right]^{\dagger}.C \tag{1.8}$$

$$A/_B\boldsymbol{C} = A.\begin{bmatrix} C^T & B^T \end{bmatrix}.\left(\begin{bmatrix} C.C^T & C.B^T \\ B.C^T & B.B^T \end{bmatrix}^{\dagger}\right)_{primeras\ r\ columnas}.C \tag{1.9}$$

1.2. Descomposiciones matriciales

Las descomposiciones matriciales permiten implementar algoritmos de una forma eficiente, reduciendo el costo computacional de los algoritmos. En este proyecto se utilizaran la descomposición QR y la descomposición en valores singulares, las cuales se describiran en la siguiente sección.

1.2.1. Descomposición RQ

Se define a H como la concatenacion de los bloques de Hankel de entrada y de salida así:

$$H = \frac{1}{\sqrt{j}}\begin{bmatrix} U_p \\ U_f \\ Y_p \\ Y_f \end{bmatrix} = R.Q^T$$

Donde $H \in \mathbb{R}^{2(m+l)ixj}$, $Q^T in \mathbb{R}^{2(m+l)ixj}$ es una matriz ortonormal ($Q.Q^T = I_{2(m+l).i}$) y $R \in \mathbb{R}^{2(m+l).i\ x\ 2(m+l).i}$ es una matraz triangular inferior.
La principal ventaja de la descomposición QR, es que R contiene la información relevante de la matriz descompuesta. Esto implica que no necesariamente se debe calcular la matriz Q reduciendo de esta manera reducir significativamente el costo computacional de los algoritmos.

1.2.2. Proyección ortogonal utilizando la descomposición RQ

La proyeccion ortogonal $A/\boldsymbol{B}^{\perp}$ puede ser expresada en función de la descomposición RQ. Definiendo A y B a través de la descomposición QR:

$$A = R_A.Q^T$$
$$B = R_B.Q^T$$

entonces, se tiene:

$$\begin{aligned} A/\boldsymbol{B}^{\perp} &= A\prod_{B^{\perp}} \\ &= \left[R_A.Q^T.Q.R_B^T\right].\left[R_B.Q^T.Q.R_B^T\right]^{\dagger}.R_B.Q^T \\ &= R_A.R_B^T\left[R_B.R_B^T\right]^{\dagger}.R_B.Q^T \end{aligned}$$

1.2.3. Proyección oblicua utilizando la descomposición RQ

En la ecuaciación (1.8) se presento la definición de la proyeccion oblicua:

$$\begin{aligned} A/_B\boldsymbol{C} &= \left[A/\boldsymbol{B^T}\right].\left[C/\boldsymbol{B^T}\right]^{\dagger}.C \\ &= R_A.\left[I_{2(m+l).i} - R_B^T\left[R_B : RB^T\right]^{\dagger}.R_B\right] \\ &\quad x\left[R_C\left[I_{2(l+m).i} - R_B^T.\left[R_B.R_B^T\right]^{\dagger}.R_B\right]\right]^{\dagger}.R_C.Q^T \end{aligned} \tag{1.10}$$

1.2.4. Proyección oblicua utilizando la teoría de mínimos cuadrados

En la sección 1.1.2 se definió la proyección oblicua como la descomposición de una matriz A a través de 2 matrices no ortogonales B y C y sus complementos ortogonales $B^{\perp}$ y $C^{\perp}$, matemáticamente esto se puede representar de la siguiente forma:

$$A = L_B.B + L_C.C + L_{B^{\perp},C^{\perp}}.\begin{bmatrix}B\\C\end{bmatrix}^{\perp}$$

La matriz $L_C.C$ se define como la proyección oblicua del espacio de filas de A a lo largo del espacio de filas de B sobre el espacio de filas de C:

$$A/_B\mathbf{C} \overset{def}{=} L_C.C.$$

La proyección oblicua $O_i = Y_f/_{U_f}\mathbf{W_p}$ puede ser calculada utilizando el algoritmo de mínimos cuadrados:

$$\left(\hat{L_u}\ \hat{L_y}\right) = \underset{L_u,L_y}{\operatorname{argmin}}\left\|\left[L_uL_y\right]\left[\begin{bmatrix}U_p\\U_f\end{bmatrix}Y_p\right] - Y_f\right\|$$

$$O_i = \hat{L_u}\left(:, 1:im\right)U_p + \hat{L_y}Y_p$$

1.3. Matrices de Hankel y secuencias de estado

Las matrices de Hankel desempeñan un papel importante en la teoria de identificación de subespacios. Estos se pueden construir a partir de los datos de entrada y salida y se definnen así:

$$U_p = \begin{bmatrix} u_0 & u_1 & \dots & u_{j-1} \\ u_1 & u_2 & \dots & u_j \\ \vdots & \vdots & \dots & \vdots \\ u_{i-1} & u_i & \dots & u_{i+j-2} \end{bmatrix} \qquad U_f = \begin{bmatrix} u_i & u_{i+1} & \dots & u_{i+j-1} \\ u_i+1 & u_{i+2} & \dots & u_{i+j} \\ \vdots & \vdots & \dots & \vdots \\ u_{2i-1} & u_{2i} & \dots & u_{2i+j-2} \end{bmatrix}$$

$$U_p^+ = \begin{bmatrix} u_0 & u_1 & \dots & u_{j-1} \\ u_1 & u_2 & \dots & u_j \\ \vdots & \vdots & \dots & \vdots \\ u_{i-1} & u_i & \dots & u_{i+j-2} \\ u_i & u_i+1 & \dots & u_{i+j-1} \end{bmatrix} \qquad U_f^- = \begin{bmatrix} u_{i+1} & u_{i+2} & \dots & u_{i+j} \\ \vdots & \vdots & \dots & \vdots \\ u_{2i-1} & u_{2i} & \dots & u_{2i+j-2} \end{bmatrix}$$

Los bloques de matrices de Hankel para los datos de salida $Y_p,\ Y_f,\ Y_p^+$ y Y_f^- son definidas de forma similar. Los bloques de Hankel de las entradas y las salidas se pueden concatener un una sola matriz

de la siguiente forma:

$$W_p = \begin{bmatrix} U_p \\ Y_p \end{bmatrix} \qquad\qquad W_f = \begin{bmatrix} U_f \\ Y_f \end{bmatrix}$$

De similar manera se puede definir W_p^+:

$$W_p^+ = \begin{bmatrix} U_p^+ \\ Y_p^+ \end{bmatrix}$$

La secuencia de estados es $X_i \in \mathbb{R}^{nxj}$ se define así:

$$X_i = \begin{bmatrix} x_i & x_{i+1} & \dots & x_{i+j-2} & x_{i+j-1} \end{bmatrix}$$

Dónde:

- El número de filas es definido por el usuario, el cual debe ser por lo menos tan grande como el máximo orden del sistema que se desea identificar.
- El número de columnas es típicamente igual a N, donde N es el número de datos muestreados, lo cual implica el uso de todos los datos

1.3.1. Matrices relacionadas con el sistema

Los algoritmos de identificación por subespacios hacen un uso extensivo de las matrices de observabilidad y controlabilidad y sus estructuras. La extensión (Γ_i) de la matriz de observabilidad (donde el subíndice denota el número de filas del bloque) es definido como:

$$\Gamma_i = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{pmatrix} \in \mathbb{R}^{li \text{ x } n}$$

Se asume que la pareja A, C son observables, lo cual implica que el rango de Γ es igual a n . La inversa de la extensión de la matriz de observabilidad Δ es definida como:

$$\Delta_i^d = \begin{pmatrix} A^{i-1} & A^{i-2}B & \dots & AB & B \end{pmatrix} \in \mathbb{R}^{n \text{ x } mi}$$

Se asume que la pareja de matrices A, B son observables. El modo de control puede ser estable o inestable. La matriz triangular inferior de Toeplitz H_i^d es definida como:

$$H_i = \begin{pmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ CA^{i-2}B & CA^{i-4}B & D & \dots & D \end{pmatrix} \in \mathbb{R}^{li \text{ x } mi}$$

1.4. Propiedades geométricas

1.4.1. Ecuaciones matriciales de entrada-salida

El siguiente teorema establece que la ecuación 1.1 puede ser reformulada en una forma matricial.

Teorema* 1 *Ecuaciones matriciales de entrada-salida

$$Y_p = \Gamma_i X_p + H_i U_p \tag{1.11}$$
$$Y_f = \Gamma_i X_f + H_i U_f \tag{1.12}$$
$$X_f = A^i X_p + \Delta_i U_p \tag{1.13}$$

La interpretación geométrica de la ecuación 1.12 es ilustrada en la figura 1.3

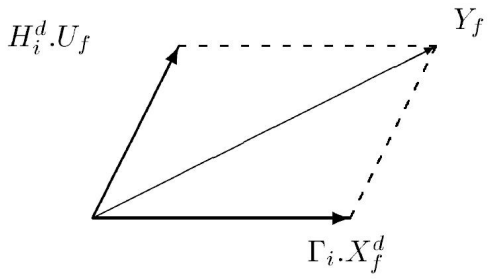

Figura 1.3: Interpretación geométrica de la ecuación 1.12

Teorema* 2 *: Teorema principal
Los vectores en el espacio de filas del bloque de Hankel Y_f son obtenidos como una suma de combinaciones lineales de vectores en el espacio de filas de la secuencia de estados X_f y combinaciones lineales de vectores en el espacio de filas del bloque de Hankel U_f.

Las consecuencias del teorema de identificación determinista son las siguientes:

- La secuencia de estados puede ser determinado directamente de los datos de entrada y salida sin necesidad de conocer el sistema de matrices .
- La matriz de observabilidad extendida puede ser calculado directamente de los datos de entrada y salida del sistema.

Definición* 1 *: Persistencia de excitación

La secuencia de entrada $u_k \in \mathbb{R}^m$ es persitentemente excitante si la covarianza de la matriz de entrada

$$R^{uu} = cov[U_{0|2i-1}, U_{0|2i-1}]$$

tiene rango completo, es decir $2.m.i$.

Teorema* 3 *:Identificación determinista:

Suponiendo que:

- La entrada u_k tiene una percistencia de excitación de orden $2i$

- La intersección del espacio de filas de U_f y el espacio de filas de X_p es vacía.
- Las matrices de ponderación $W_1 \in \mathbb{R}^{li \text{ x } li}$ y $W_2 \in \mathbb{R}^{j \text{ x } j}$, definidas por el usuario son tales que $W1$ es de rango completo y $W2$ cumple que: $rango(W_p) = rango(W_p.W_2)$, donde W_p es el bloque de hankel que concatena las entradas y salidas del sistema.

Y con $\mathcal{O}_i$ definida como la proyección oblicua:

$$\mathcal{O}_i \stackrel{def}{=} Y_f/_{U_f}\mathbf{W_p}$$

Y la descomposición en valores singulares:

$$W_1.\mathcal{O}_i.W_2 = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} \tag{1.14}$$

Se tiene:

1. La matriz $\mathcal{O}_i$ es equivalente al producto de la matriz de observabilidad extendidad Γ_i y los estados X_f:
$$\mathcal{O}_i = \Gamma_i.X_f. \tag{1.15}$$
2. El orden del sistema 1.1 se puede determinar a partir de los valores singulares en la ecuación 1.14
3. La matriz de observabilidad extendida Γ_i es igual a:
$$\Gamma_i = W_1^{-1}U_1S_1^{1/2}$$
4. La secuencia de estados X_f es igual a :
$$X_f = \Gamma_i^{\dagger}.\mathcal{O}_i \tag{1.16}$$

1.5. Algoritmos de identificación N4SID y CCA

En el teorema 3 se introdujeron las matrices de ponderación W_1 y W_2. La elección especifica de dichas matrices conducen a diferentes algoritmos de identificación el cual determina el modelo de espacio de estados que se puede obtener.

1.5.1. N4SID

El acrónimo N4SID significa "**N**umerical algorithms for **S**ubspace **S**tate **S**pace **S**ystem **I**dentification ". Este algoritmo determina el orden del sistema y la matriz de observabilidad extendida Γ directamente de la descomposición en valores singulares de la proyección oblicua:

$$\mathcal{O}_i = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} \tag{1.17}$$

El orden del sistema es igual al numero de valores singulares en S_1 diferentes de cero. La matriz de observabilidad extendida es calculada de la siguiente manera:

$$\Gamma_i = U_1.(S_1)^{1/2}$$

Este algoritmo fácilmente puede ser descrito por el teorema 3, usando las siguientes matrices de ponderación:

$$W_1 = I_{li},$$
$$W_2 = I_j$$

1.5.2. CCA

en este algoritmo, la elección de las matrices de ponderación W_1 y W_2,se seleccionan con base en el análisis de variación canónica (CVA). Específicamente se considera la correlación canónica entre el pasado W_p condicionado a las entradas futuras U_f y las salidas futuras Y_f condicionadas a las entradas futuras U_f.Matemáticamente, se consideran los ángulos principales y direcciones entre $W_p/U_f^{\perp}$ y $Y_f/U_f^{\perp}$.

En [3] se muestra como el orden del sistema es igual al número de ángulos principales diferentes de $\pi/2$ en $[W_p/U_f^{\perp} \angle Y_f/U_f^{\perp}]$ y se muestra que el algoritmo de análisis de correlación canónica (CCA) corresponde al teorema 3, con las matrices de ponderación W_1 y W_2 definidas como:

$$W_1 = cov(Y_f/U_f^{\perp}, Y_f/U_f^{\perp})$$
$$W_2 = \Pi_{U_f^{\perp}}$$

La anterior definición para W_1 y W_2 hace que los valores singulares en la ecuación1.14 correspondan a los cosenos de los ángulos principales $[W_p/U_f^{\perp} \angle Y_f/U_f^{\perp}]$

1.6. Algoritmos de identificación de sistemas lineales

En esta sección se presentara un resumen de los algoritmos de identificación por subespacios de estado N4SID y CCA:

1. Se calcula la proyección oblicua:

$$O_i = Y_f/_{U_f}\boldsymbol{W_p}$$
$$O_{i-1} = Y_f^-/_{U_f^-}\boldsymbol{W_p^+}$$

2. Se descompone la proyección oblicua O_i

$$W_1.O_i.W_2 = USV^T$$

Donde las matrices W_1 y W_2 dependen del algoritmo de identificación:

Algoritmo	$\mathbf{W_1}$	$\mathbf{W_2}$
N4SID	I_{li}	I_j
CCA	$W_1 = cov(Y_f/U_f^{\perp}, Y_f/U_f^{\perp})$	$W_2 = \Pi_{U_f^{\perp}}$

3. Determinamos el orden n del sistema a partir de los valores singulares S de la descomposición en valores singulares de O_i y se obtiene S_1 y U_1.

4. Determinar Γ_i y Γ_{i-1} de la siguiente forma:

$$O_i = USV^T$$

5. Determinar X_i y X_{i+1}:

$$X_i = \Gamma_i^{\dagger}.O_i$$
$$X_{i+1} = \Gamma_{i-1}^{\dagger}.O_{i-1}$$

6. Determinar A, B, C y D a partir del siguiente sistema de ecuaciones:

$$\begin{bmatrix} X_{i+1} \\ Y_i \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} X_i \\ U_i \end{bmatrix}$$

Capítulo 2

IDENTIFICACIÓN DE SISTEMAS NO LINEALES

2.1. Modelos Hammerstein

El modelo Hammerstein es un sistema de bloques orientados que permiten solucionar el problema de la no linealidad en la identificación de modelos no lineales a través de una no linealidad estática $f : \mathbb{R}^m \longrightarrow \mathbb{R}^m$, Con esta definición para la no linealidad f el sistema en (1.1) puede ser re-definido así:

$$\begin{array}{c} x_{t+1} = A.x_t + B.f\left(u_t\right) \\ y_t = C.x_t + D.f\left(u_t\right) \end{array} \tag{2.1}$$

2.2. Modelo Hammerstein-Wiener

El modelo Hammerstein-Wiener al igual que el modelo Hammerstein, es un sistema de bloques orientados que a través de dos no linealidades estáticas $f : \mathbb{R}^m \longrightarrow \mathbb{R}^m$ y $g : \mathbb{R}^l \longrightarrow \mathbb{R}^l$ permite solucionar el problema de la no linealidad en la identificación de modelos no lineales Con esta definición para las no linealidades f y g y asumiendo que $g^(-1)$ existe para todos los posibles valores de las salidas, el sistema en (1.1) puede ser re-definido así:

$$\begin{array}{c} x_{t+1} = A.x_t + B.f\left(u_t\right) \\ g^{-1}\left(y_t\right) = C.x_t + D.f\left(u_t\right) \end{array} \tag{2.2}$$

2.3. Función regresión LS-SVM

Tenemos un conjunto de datos de entrenamiento $\{x_t, y_t\}_{t=0}^{N-1}$ donde $x_t \in \mathbb{R}^d$ y $y_t \in \mathbb{R}$, con x_t como entrada y y_t. Considere el modelo de regresión $y_t = f\left(x_t\right) + e_t$, donde f es una función desconocida $f : \mathbb{R}_d \longrightarrow \mathbb{R}$ y $e_0, e_1, \ldots, e_{N-1}$ son errores aleatorios no relacionados.
Se asume el siguiente modelo:

$$f\left(x\right) = w^T \varphi\left(x\right) + b$$

Donde φ es una función no lineal que mapea los datos de entrenamiento a un espacio de mayor dimensión n_H, $\varphi : \mathbb{R}^d \longrightarrow \mathbb{R}^{n_H}$.

La funcion de costo del LS-SSVM

$$\min_{w,b,e} \mathbb{J}(w,e) = \frac{1}{2}w^T w + \frac{\gamma}{2}\sum_{t=1}^{n} e_t^2$$
$$s.a: \; y_t = w^t \varphi(x_t) + b + e_t, t = 0,1,\ldots,N-1.$$

γ: constante positiva de regularización. utilizando el lagrangiano :

$$\mathcal{L}(w,b,e,\alpha) = \mathbb{J} - \sum_{t=0}^{N-1} \alpha_t \left\{ w^T \varphi(x_t) + b + e_t - y_t \right\}$$

Donde α son los multiplicadores de Lagrange. Aplicando las condiciones de optimalidad $\frac{\partial \mathcal{L}}{\partial w} = 0$, $\frac{\partial \mathcal{L}}{\partial b} = 0$, $\frac{\partial \mathcal{L}}{\partial e_t} = 0$ y $\frac{\partial \mathcal{L}}{\partial \alpha_t} = 0$, se obtiene el siguiente sistema de ecuaciones:

$$\begin{bmatrix} 0 & 1_N^T \\ 1_N & \Omega + \gamma^{-1} I_N \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{2.3}$$

Resolviendo el anterior sistema podemos obtener (b,α) y $\hat{f}$:

$$\hat{f}(x_\oplus) = \sum_{t=0}^{N-1} \alpha_t K(x_\oplus, x_t) + b$$

Donde $y = \{y_0 \quad y_1 \quad \ldots \quad y_{N-1}\}$, $1_N = \{1 \quad 1 \quad \ldots \quad 1\} \in \mathbb{R}^N$, $\alpha = \{\alpha_0 \quad \alpha_1 \quad \ldots \quad \alpha_{N-1}\}$, $\Omega_{i,j} = K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$, $\forall i,j = 0,1,\ldots,N-1$, donde K es un kernel simétrico y definido positivo.

2.4. Extensión del algoritmo N4SID para la identificación de un modelo Hammerstein

En la sección 2.1 se presento el sistema Hammerstein, en el cual se introdujo la función no lineal f la cual transforma la ecuación 1.1 en 2.1. Se define Φ como un operador sobre un bloque de Hankel, la cual se aplica sobre cada bloque Z_i de $Z \in \mathbb{R}^{qmxp}$:

$$\Phi_{\mathcal{F}} \left(\begin{bmatrix} Z_1 & Z_2 \ldots & Z_p \\ Z_2 & Z_3 \ldots & Z_{p+1} \\ \vdots & \vdots & \vdots \\ Z_q & Z_{q+1} \ldots & Z_{p+q-1} \end{bmatrix} \right) = \begin{bmatrix} \mathcal{F}(Z_1) & \mathcal{F}(Z_2) \ldots & \mathcal{F}(Z_p) \\ \mathcal{F}(Z_2) & \mathcal{F}(Z_3) \ldots & \mathcal{F}(Z_{p+1}) \\ \vdots & \vdots & \vdots \\ \mathcal{F}(Z_q) & \mathcal{F}(Z_{q+1}) \ldots & \mathcal{F}(Z_{p+q-1}) \end{bmatrix}$$

2.4.1. Proyección oblicua $\mathcal{O}_i$

La proyección oblicua $O_i = Y_f /_{U_f} \mathbf{W_p}$ puede ser calculada obteniendo L_u, L_y y f obtenidas a partir de la minimización de E de la siguiente ecuación:

$$Y_f = \begin{bmatrix} L_u & L_y \end{bmatrix} \begin{bmatrix} \Phi(U_{0|2i-1}) \\ Y_p \end{bmatrix} + E \tag{2.4}$$

Esto puede ser reescrito así:

$$Y_f(s,t) = L_Y(s,:)Y_p(:,t) + \sum_{h=1}^{2i} L_u(s,(h-1)m+1:hm) f(U_{h+t-2}) + E(s,t) \tag{2.5}$$
$$\forall \; s = 1,\ldots,il; t = 1,\ldots,j$$

Una vez estimados $\hat{L_u}$, $\hat{L_Y}$ y $\hat{f}$ a través de 2.4 y 2.5, la proyección oblicua es calculada así:

$$O_i(s,t) = \hat{L_y}(s,:)Y_p(:,t) + \sum_{h=1}^{i} \hat{L_u}(s,(h-1)m++1:hm)\hat{f}(u_{h+t-2}) \tag{2.6}$$

En 2.5 y2.6 aparecen productos de $\hat{L_u}$, $\hat{L_y}$ y $\hat{f}$, lo cual aumenta la complejidad en un problema de optimización, para resolver esto introduciremos un conjunto de funciones $g_{h,s} : \mathbb{R}^m \longrightarrow \mathbb{R}$.

$$g_{h,s} \stackrel{def}{=} C_{h,s}^T.f = L_u(s,(h-1)m+1:hm) \tag{2.7}$$

para $h = 1,\ldots,2i$ y $1,\ldots,il$. Con estas nuevas funciones se puede obtener una generalización para 2.5 y 2.6:

$$Y_f(s,t) = L_y(s,:)Y_p(:,t) + \sum_{h=1}^{2i} g_{h,s}(u_{h+t-2}) + E(s,t) \tag{2.8}$$

$$\mathcal{O}_i = L_y\hat{(}s,:)Y_p(:,t) + \sum_{h=1}^{i} \hat{g}_{h,s}(u_{h+t-2}) \tag{2.9}$$

$\hat{L_y}$ puede ser estimado utilizando la teoría LS-SVM (revisar 2.3). Sustituyendo $g_{h,s}$ por el modelo primal $w_{h,s}^T$:

$$Y_f(s,t) = L_y(s,:)Y_p(:,t) + \sum_{h=1}^{2i} w_{h,s}^T\phi(u_{h+t-2}) + E(s,t) \tag{2.10}$$
$$\forall s = 1,\ldots,li; \quad t = 1,\ldots,j$$

El problema para la LS-SVM es formulado a través de un problema de optimizacion con restricciones:

$$\begin{aligned} \min_{w_{h,s},L_y,E} \mathbb{J}(w_{h,s},L_y,E) &= \frac{1}{2}\sum_{s=1}^{il}\sum_{h=1}^{2i} w_{h,s}^T w_{h,s} + \frac{\lambda}{2}\sum_{s=1}^{il}\sum_{t=1}^{j} E(s,t)^2 \\ s.a \begin{cases} Y_f(s,t) = L_y(s,:)Y_p(:,t) + \sum_{h=1}^{2i} w_{h,s}^T\phi(u_{h+t-2}) + E(s,t) \quad (a) \\ \forall s = 1,\ldots,li; \quad t = 1,\ldots,j \end{cases} \end{aligned} \tag{2.11}$$

Lema 3.1: Dado el problema primal en 2.11, estimamos L_y y $\mathcal{A}$ del siguiente sistema:

$$\begin{bmatrix} 0 & Y_p \\ Y_p^T & \mathcal{K} + \gamma^{-1}.I \end{bmatrix} \begin{bmatrix} L_y^T \\ \mathcal{A} \end{bmatrix} = \begin{bmatrix} 0 \\ Y_f^T \end{bmatrix}$$

donde

$$\mathcal{A} = \begin{bmatrix} \alpha_{1,1} & \ldots & \alpha_{li,1} \\ \vdots & \vdots & \vdots \\ \alpha_{1,j} & \ldots & \alpha_{li,j} \end{bmatrix}$$

$$\mathcal{K}(p,q) = \sum_{h=1}^{2i} K(u_{h+p-2}, u_{h+p-2})$$

para todo $p,q = 1,\ldots,j$. La estimación para $g_{h,s}$ en 2.8 esta dada por:

$$\hat{g}_{h,s}(u^*) = \alpha_{h,s}\sum_{t=1}^{j} K(u_t,u^*) \tag{2.12}$$
$$\forall h,s$$

combinando los resultados del lema 3.1 con 2.6, se puede estimar la proyección oblicua:

$$\mathcal{O}_i = \mathcal{A}^T \mathcal{K}_p + \hat{L_y} Y_p \tag{2.13}$$

$$\text{con } \mathcal{K}_p = \sum_{h=1}^{i} K(u_{h+p-2}, u_{h+q-2}) \text{ para todo } p, q = 1, \ldots, j.$$

2.4.2. Proyección oblicua $\mathcal{O}_{i+1}$

El calculo de $\mathcal{O}_{i+1}$ es totalmente equivalente al de $\mathcal{O}_i$:

$$\mathcal{O}_{i+1} = (\mathcal{A}^-)^T (K_p^+)^T + L_y^- Y_p^+ \tag{2.14}$$

para todo $p, q = 1, \ldots, j$.

$\mathcal{A}^-$ y L_y^- se calculan del siguiente sistema:

$$\begin{bmatrix} 0 & Y_p^+ \\ (Y_p^+)^T & \mathcal{K} + \gamma^{-1}.I \end{bmatrix} \begin{bmatrix} (L_y^T)^T \\ \mathcal{A}^- \end{bmatrix} = \begin{bmatrix} 0 \\ (Y_f^-)^T \end{bmatrix}$$

2.4.3. Estimación de los estados

Las secuencias de estado X_i y X_{i+1} pueden ser calculados de forma análoga a lo hecho en la sección 1.6. Estas secuencias de estado pueden ser calculadas en un segundo paso del algoritmo para estimar el sistema de matrices y la función no lineal f.

2.4.4. Extracción del sistema de matrices y la función no lineal f

El modelo lineal y la función f pueden ser estimados de:

$$\begin{aligned} &(\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{f}) \\ &= \min_{A,B,C,D,f} \left\| \begin{bmatrix} X_{i+1} \\ Y_{i\ i} \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_i \\ \Phi(Ui\ i) \end{bmatrix} \right\|_F^2 \end{aligned} \tag{2.15}$$

Este problema puede ser re escrito como un problema LS-SVM. Denotando

$$\mathcal{X}_{i+1} = \begin{bmatrix} X_{i+1} \\ Y_{i|i} \end{bmatrix} \quad \Theta_{AC} = \begin{bmatrix} A \\ C \end{bmatrix} \quad \Theta_{BD} = \begin{bmatrix} B \\ D \end{bmatrix} \tag{2.16}$$

$$\mathcal{X}_{i+1} = \Theta_{AC} X_i + \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_{n+l}^T \end{bmatrix} \Phi_\phi(U_{i\ i}) + E \tag{2.17}$$

Donde E es el residuo de 2.15. El resultado del problema primal LS-SVM puede ser escrito como:

$$\min_{w_s, E, \Theta_{AC}} \mathbb{J}(w, E) = \frac{1}{2} \sum_{s=1}^{n+l} w_s^T w_s + \frac{\lambda_{BD}}{2} \sum_{s=1}^{il} \sum_{t=1}^{j} E(s,t)^2$$

$$s.a \begin{cases} \mathcal{X}_{i+1}(s,t) = \Theta_{AC}(s,:) X_i(:,t) + w_s^T \phi(u_{i+t-1}) \quad (a) \\ \forall s = 1, \ldots, li; \quad t = 1, \ldots, j \end{cases} \tag{2.18}$$

Donde γ_{BD} es una constante de regularización.

LEMA 2.2 La estimación para A y C en Θ_{AC} es obtenido del siguiente problema dual:

$$\begin{bmatrix} 0 & X_i \\ X_i^T & \mathcal{K}_{BD} + \gamma_{BD}^{-1} I \end{bmatrix} \begin{bmatrix} \Theta_{AC}^T \\ \mathcal{A}_{BD} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathcal{X}_{i+1} \end{bmatrix} \tag{2.19}$$

Por lo cual $w_s = \sum_{t=1}^{j} \alpha_{s,t}\phi(u_{i+t-1}) + \beta_s \sum_{t=0}^{N-1} \phi(u_t)$, para todo $s = 1, \ldots, n+l$, $\mathcal{K}_{BD}(p,q) = K(u_{i+p-1}, u_{i+q-1})$ para todo $(p,q) = 1, \ldots, j$ y

$$\mathcal{A} = \begin{bmatrix} \alpha_{1,1} & \ldots & \alpha_{n+l,1} \\ \vdots & \vdots & \vdots \\ \alpha_{1,j} & \ldots & \alpha_{n+l,j} \end{bmatrix}$$

Combinando los resultados del lema 2.2 con 2.16 y 2.15, se tiene:

$$\Theta_{BD}[f(u_0)\ f(u_1)\ \ldots\ f(u_{N-1})] = \mathcal{A}_{BD}^T \Omega(i+1 : i+j, :) \tag{2.20}$$

2.5. Identificación de un modelo Hammerstein-Wiener

Figura 2.1: Estructura del modelo Hammerstein-Wiener

Donde $w(t) = f(u(t))$ es una función de transformación no lineal de los datos de entrada. $w(t)$ tiene la misma dimensión de $u(t)$.$x(t) = (B/F)w(t)$ es una función de transferencia lineal. $x(t)$ tiene la misma dimensión de $y(t)$. El modelo Hammerstein-Wiener calcula la salida y en tres pasos:

- Calcular $w(t)$ de los datos de entradas. $w(t)$ es la entrada de la función de transferencia B/F. La entrada no lineal es una función estática, donde el valor de la salida en un dado tiempo t solo depende del valor de entrada en el tiempo t.
- Calcular la salida del bloque lineal usando $w(t)$ y las condiciones iniciales x(t)=(B/F)w(t). El bloque lineal puede ser configurado a través de la especificación del orden del numerador B y el denominador F.
- Calcular la salida del modelo usando la función no lineal $h : y(t) = h(x(t))$ para transformar la salida del bloque lineal $x(t)$.

La estimación de los parámetros del modelo se hacen a través de un proceso iterativo, en donde cada iteración consta de dos pasos: identificar el modelo paramétrico de un sistema lineal y la solución de un problema lineal de mínimos cuadrados para identificar las no linealidades estáticas [4].

Capítulo 3

Resultados

En este capitulo se presentan los resultados de la implementación de los algoritmos de identificación desarrollados en los capítulos anteriores. Para los algoritmos de identificación lineal se utilizan modelos generados de forma aleatoria por la función de matlab *drss(n,p,m)* donde n representa el orden del sistema, p representa el número de salidas del sistema y m representa el número de entradas del sistema.

3.1. Implementación del algoritmo N4SID

3.1.1. CASO 1: Modelo SISO de segundo orden

Modelo generado

$$A = \begin{bmatrix} 0{,}075652 & -0{,}1573 \\ 0{,}1573 & 0{,}07651 \end{bmatrix}$$
$$B = \begin{bmatrix} 0{,}2886 \\ -0{,}765 \end{bmatrix}$$
$$C = \begin{bmatrix} 0{,}9513 & 0 \end{bmatrix}$$
$$D = \begin{bmatrix} 0{,}2323 \end{bmatrix}$$

Modelo estimado

$$A_{estimado} = \begin{bmatrix} 0{,}3159 & 0{,}1261 \\ -0{,}6506 & -0{,}1629 \end{bmatrix}$$
$$B_{est} = \begin{bmatrix} -0{,}2619 \\ -0{,}2456 \end{bmatrix}$$
$$C_{est} = \begin{bmatrix} -1{,}0964 & 0{,}0513 \end{bmatrix}$$
$$D_{est} = \begin{bmatrix} 0{,}2323 \end{bmatrix}$$

En la figura 3.1 se pueden observar los resultados de la identificación.

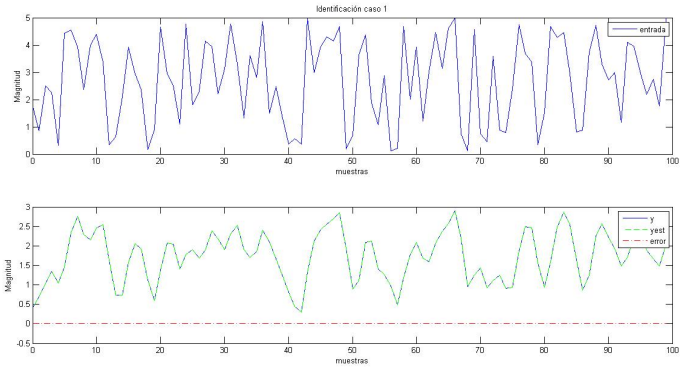

Figura 3.1: Identificación del modelo siso

3.1.2. CASO 2: Modelo MISO de orden 2 con 2 entradas y 1 salida.

Modelo generado

$$A = \begin{bmatrix} -0{,}6113 & -0{,}3475 \\ -0{,}3475 & -0{,}5855 \end{bmatrix}$$

$$B = \begin{bmatrix} 0{,}1718 & 0 \\ -0{,}3287 & -0{,}1783 \end{bmatrix}$$

$$C = \begin{bmatrix} 0{,}6545 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} -0{,}4354 & 0 \end{bmatrix}$$

Modelo estimado

$$A_{est} = \begin{bmatrix} -0{,}9569 & 0{,}1322 \\ -0{,}0574 & -0{,}2400 \end{bmatrix}$$

$$B_{est} = \begin{bmatrix} -0{,}4864 & 0{,}1469 \\ 0{,}1323 & -0{,}1757 \end{bmatrix}$$

$$C_{est} = \begin{bmatrix} -0{,}2991 & 0{,}2500 \end{bmatrix}$$

$$D_{est} = \begin{bmatrix} -0{,}4254 & 0 \end{bmatrix}$$

En la figura 3.2 se puede observar las entradas y en la figura 3.3 se puede observar el desempeño de la identificación.

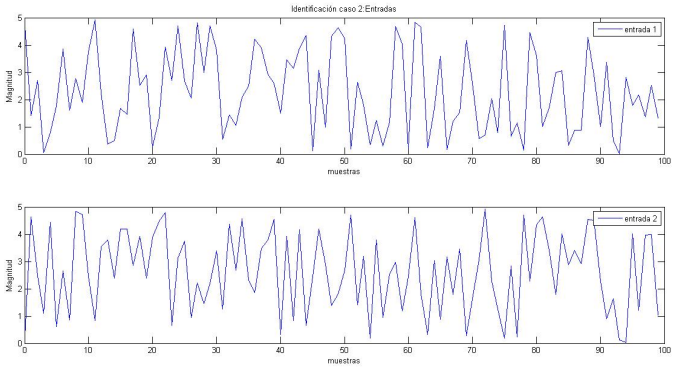

Figura 3.2: Entradas del sistema miso

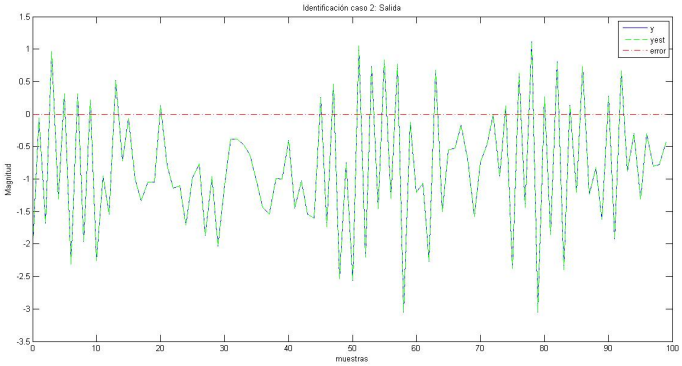

Figura 3.3: Identificación del sistema miso

3.1.3. CASO 3: Modelo MIMO de orden 3 con 2 entradas y 2 salidas

Modelo generado

$$A = \begin{bmatrix} -0{,}1052 & 0{,}7966 & -0{,}01204 \\ -0{,}1438 & -0{,}1273 & -0{,}7835 \\ 0{,}7836 & -0{,}006358 & 0{,}02999 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}9917 & -0{,}2458 \\ 0{,}3951 & -0{,}438 \\ 0{,}8995 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} -0{,}1154 & 0{,}1255 & 0 \\ 1{,}881 & -2{,}071 & 0{,}6667 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & -0{,}9928 \end{bmatrix}$$

Modelo estimado

$$A_{est} = \begin{bmatrix} -0{,}5356 & -0{,}4203 & 0{,}0314 \\ -1{,}0578 & -0{,}1902 & -0{,}3854 \\ -0{,}7964 & 1{,}2121 & 0{,}1428 \end{bmatrix}$$

$$B_{est} = \begin{bmatrix} -1{,}6487 & ,11187 & \\ -1{,}3429 & 0{,}4456 & \\ -0{,}5700 & -0{,}68330{,}1323 & -0{,}1757 \end{bmatrix}$$

$$C_{est} = \begin{bmatrix} -0{,}1121 & 0{,}0057 & 0{,}0232 \\ 1{,}7814 & -0{,}3839 & -0{,}59190 \end{bmatrix}$$

$$D_{est} = \begin{bmatrix} 0 & 0 \\ 0 & -0{,}9928 \end{bmatrix}$$

En la figura 3.4 se puede observar las entradas y en la figura 3.5 se puede observar el desempeño de la identificación.

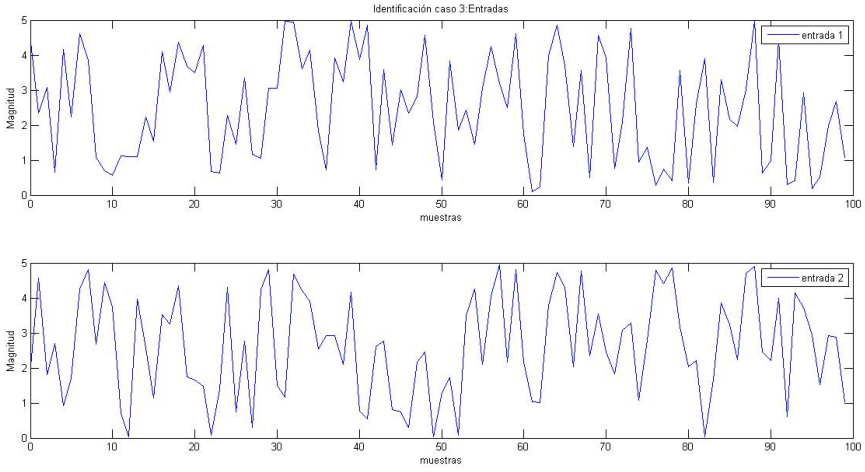

Figura 3.4: Entradas del sistema mimo

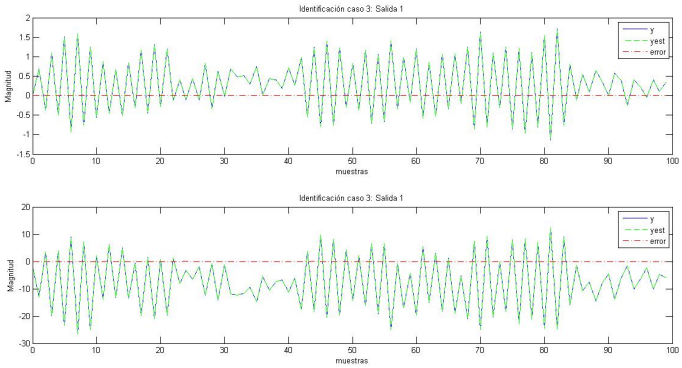

Figura 3.5: Identificación del sistema mimo

3.2. Implementación del algoritmo CCA

3.2.1. CASO 1: Modelo SISO de segundo orden

Modelo generado

$$A = \begin{bmatrix} 0{,}1864 & 0{,}1339 \\ 0{,}1339 & 0{,}4099 \end{bmatrix}$$
$$B = \begin{bmatrix} -1{,}203 \\ 1{,}038 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & -0{,}1729 \end{bmatrix}$$
$$D = \begin{bmatrix} -1{,}209 \end{bmatrix}$$

Modelo estimado

$$A_{estimado} = \begin{bmatrix} 0{,}2672 & 0{,}1678 \\ 0{,}1756 & 0{,}3290 \end{bmatrix}$$
$$B_{est} = \begin{bmatrix} -0{,}4663 \\ 0{,}1303 \end{bmatrix}$$
$$C_{est} = \begin{bmatrix} 0{,}41020{,}0908 \end{bmatrix}$$
$$D_{est} = \begin{bmatrix} -1{,}2087 \end{bmatrix}$$

En la figura 3.6 se pueden observar los resultados de la identificación.

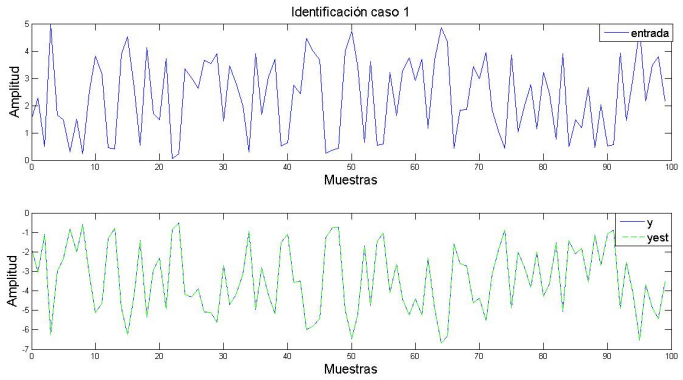

Figura 3.6: Identificación del modelo siso

3.2.2. CASO 2: Modelo MISO de orden 2 con 2 entradas y 1 salida.

Modelo generado

$$A = \begin{bmatrix} -0{,}1258 & -0{,}3791 \\ -0{,}3791 & -0{,}4692 \end{bmatrix}$$

$$B = \begin{bmatrix} 1{,}417 & -0{,}939 \\ 0{,}01148 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0{,}01697 & 0{,}2192 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Modelo estimado

$$A_{est} = \begin{bmatrix} -0{,}2386 & -0{,}1835 \\ -0{,}9250 & -0{,}3563 \end{bmatrix}$$

$$B_{est} = \begin{bmatrix} -1{,}6628 & 1{,}0980 \\ -0{,}6387 & 0{,}3997 \end{bmatrix}$$

$$C_{est} = \begin{bmatrix} 0{,}0119 - 0{,}0726 \end{bmatrix}$$

$$D_{est} = \begin{bmatrix} -1{,}407 * 10^{-16} & 3{,}1 * 10^{-18} \end{bmatrix}$$

En la figura 3.7 se puede observar las entradas y en la figura 3.8 se puede observar el desempeño de la identificación.

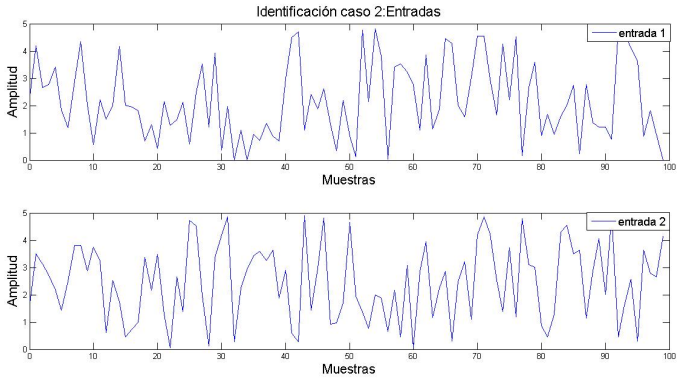

Figura 3.7: Entradas del sistema miso

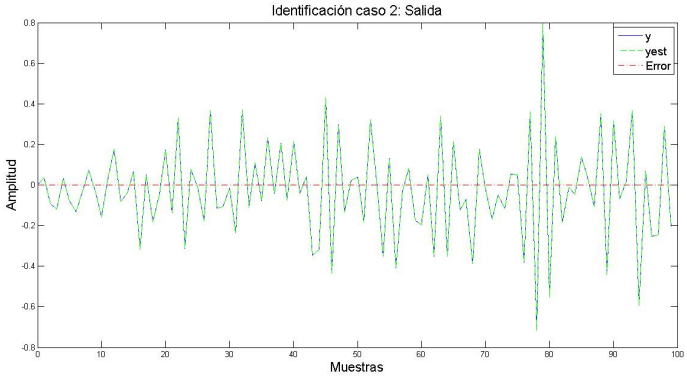

Figura 3.8: Identificación del sistema miso

3.2.3. CASO 3: Modelo MIMO de orden 3 con 2 entradas y 2 salidas

Modelo generado

$$A = \begin{bmatrix} -0{,}1052 & 0{,}7966 & -0{,}01204 \\ -0{,}1438 & -0{,}1273 & -0{,}7835 \\ 0{,}7836 & -0{,}006358 & 0{,}02999 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}9917 & -0{,}2458 \\ 0{,}3951 & -0{,}438 \\ 0{,}8995 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} -0{,}1154 & 0{,}1255 & 0 \\ 1{,}881 & -2{,}071 & 0{,}6667 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & -0{,}9928 \end{bmatrix}$$

Modelo estimado

$$A_{est} = \begin{bmatrix} -0{,}5356 & -0{,}4203 & 0{,}0314 \\ -1{,}0578 & -0{,}1902 & -0{,}3854 \\ -0{,}7964 & 1{,}2121 & 0{,}1428 \end{bmatrix}$$

$$B_{est} = \begin{bmatrix} -1{,}6487 & 0{,}11187 & \\ -1{,}3429 & 0{,}4456 & \\ -0{,}5700 & -0{,}68330{,}1323 & -0{,}1757 \end{bmatrix}$$

$$C_{est} = \begin{bmatrix} -0{,}1121 & 0{,}0057 & 0{,}0232 \\ 1{,}7814 & -0{,}3839 & -0{,}5919 \end{bmatrix}$$

$$D_{est} = \begin{bmatrix} 0 & 0 \\ 0 & -0{,}9928 \end{bmatrix}$$

En la figura 3.9 se puede observar las entradas y en la figura 3.10 se puede observar el desempeño de la identificación.

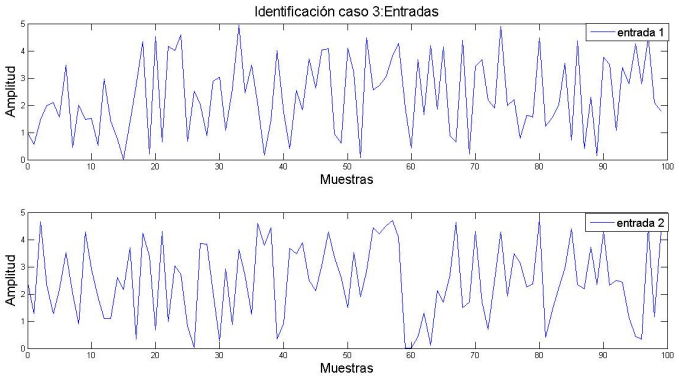

Figura 3.9: Entradas del sistema mimo

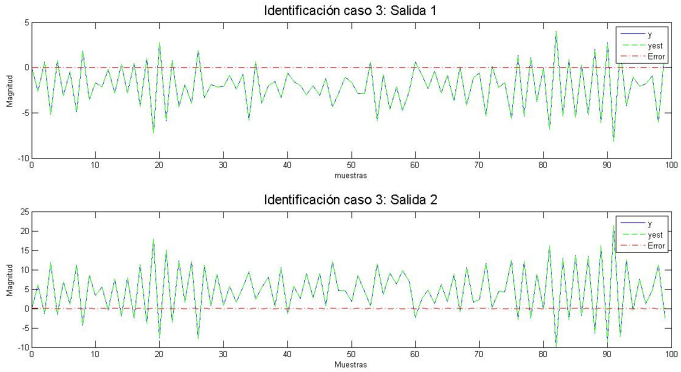

Figura 3.10: Identificación del sistema mimo

3.3. Implementación del algoritmo de identificación no lineal N4SID a traves de un modelo Hammerstein

En esta sección se implementara el algoritmo desarrollado en la sección 2.4, para lo cual se utilizaran sistemas no lineales siso y mimo.

Identificación de un sistema SISO

Considere el siguiente sistema no lineal:

$$A(Z)(y - v) = B(z)f(u) + e$$

Con A y B polinomios en el espacio Z, donde $B(z) = z^6 + 0{,}8z^5 + 0{,}3z^4 + 0{,}4z^3$ y $A(z) = (z - 0{,}98e^{\pm i})(z - 0{,}98e^{\pm 1{,}6i})(z - 0{,}97e^{\pm 0{,}4i})$ polinomios en el espacio , donde y . Sea $f = \mathbb{R} \longrightarrow \mathbb{R}$ tal que $f(u) = sinc(u).u^2$.

En la figura 3.15 se puede observar la entradas y en la figura 3.12 se puede observar el desempeño de la identificación.

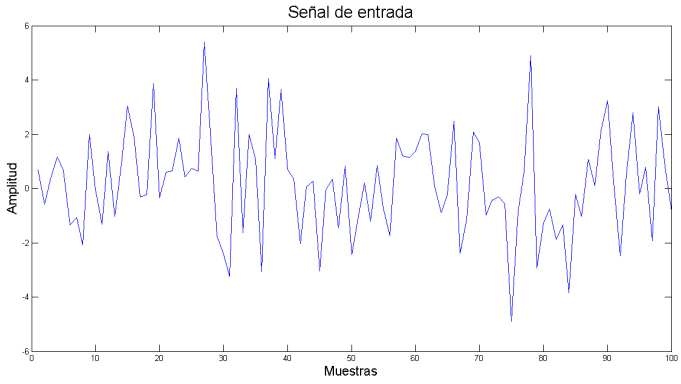

Figura 3.11: Entrada del sistema siso

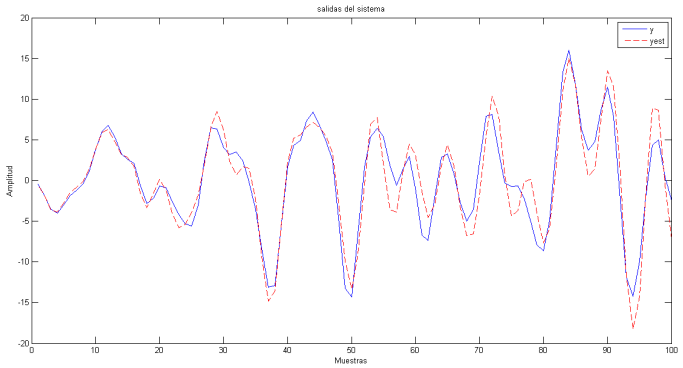

Figura 3.12: Identificación del sistema siso

Identificación de un sistema MISO

En la identificación del sistema MISO, se considera un modelo de la forma:

$$\vec{y} = H(z)f(\vec{u}) + e$$

Donde H se define así:

$$H(z) = \left[\frac{z^6+0{,}8z^5+0{,}3z^4+0{,}4z^3}{(z-0{,}98e^{\pm i})(z-0{,}98e^{\pm 1{,}6i})(z-0{,}97e^{\pm 0{,}4i})} \quad \frac{z^6+0{,}9z^5+0{,}7z^4+0{,}2z^3}{(z-0{,}98e^{\pm i})(z-0{,}98e^{\pm 1{,}6i})(z-0{,}97e^{\pm 0{,}4i})} \right]$$

Además la función vectorial $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ tiene la siguiente forma:

$$f(u) = \begin{bmatrix} sinc(u_1).u^2 \\ sinc(u_1) - sinc(u_2) \end{bmatrix} \tag{3.1}$$

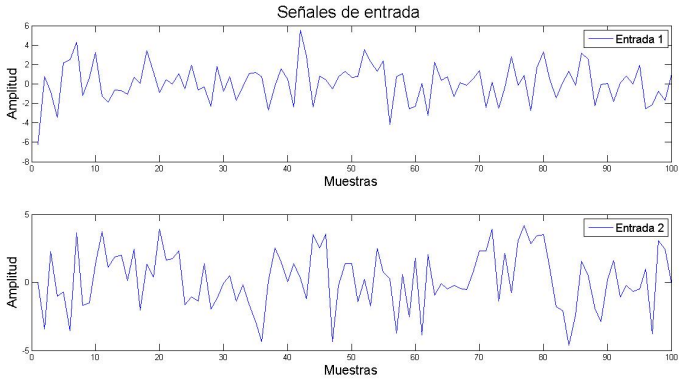

Figura 3.13: Entrada del sistema miso

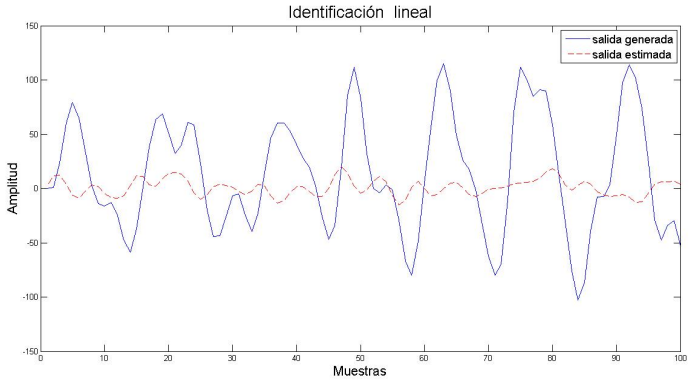

Figura 3.14: Identificación lineal: Algoritmo N4SID

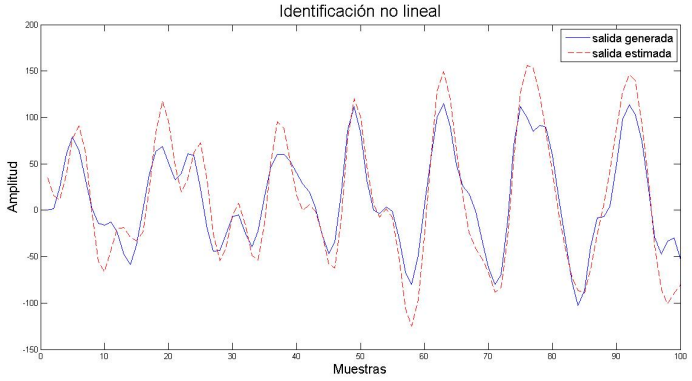

Figura 3.15: Identificación Algoritmo Hammerstein-N4SID

3.4. Identificación de sistemas de generación eólica

La identificación de generadores eólicos se hara por medio de dos sistemas de generación eólica, un sistema aislado empleando un generador asíncrono y un parque eólico conectado a la red, el cual consta de cinco turbinas acoplada cada una a un generador asíncrono.

La identificación se hará por medio del algoritmo lineal CCA y un modelo Hammerstein-wiener a través de la función *nlhw* de matlab.

Sistema de generación eólico aislado

Se presentara la identificación de un modelo de generación eólica, el cual consta de una turbina eólica de 480 V, un generador asíncrono de 275 KVA, una máquina síncrona de 480V y y 300KVA utilizada como condensador síncrono , una carga principal de 50 KW, una carga secundaría variable(0 446.25 kW) y un regulador de frecuencia.

La carga secundaria variable consta de 8 bancos de resistencias trifásicos conectados en serie a través de tiristores GTO funcionando como interruptores.

La frecuencia del sistema es controlada a través de un regulador de frecuencia que mide la frecuencia del sistema y la compara con una frecuencia de referencia para obtener un el error de frecuencia. Este error de frecuencia es integrado para obtener el error de fase. El error de fase se utiliza entonces por

un controlador proporcional-diferencial (PD) para producir una señal de salida que representa la carga secundaria requerida, esta señal es procesada y enviada al control de la carga secundaria variable para obtener la carga deseada y de esta forma regular la frecuencia.

En la figura 3.17 se puede observar la señal de entrada del sistema, en la figura 3.18 se puede observar la identificación lineal del sistema usando el algoritmo CCA y en la figura 3.23 se puede observar la identificación del sistem,usando un modelo Hammerstein-wiener.

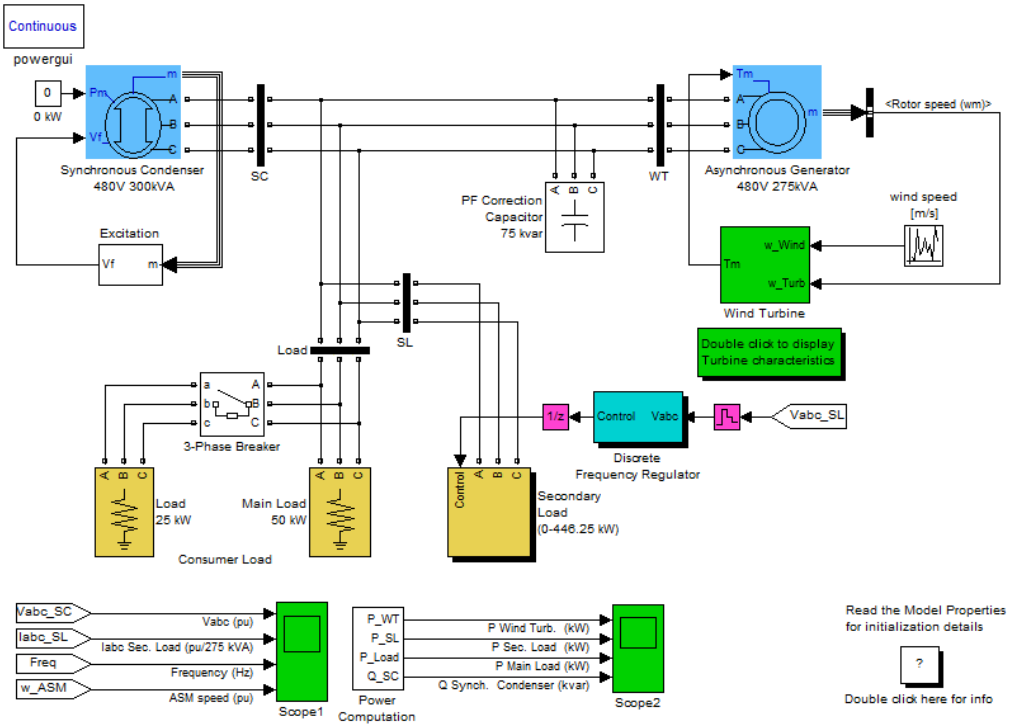

Figura 3.16: Sistema de generación aislado

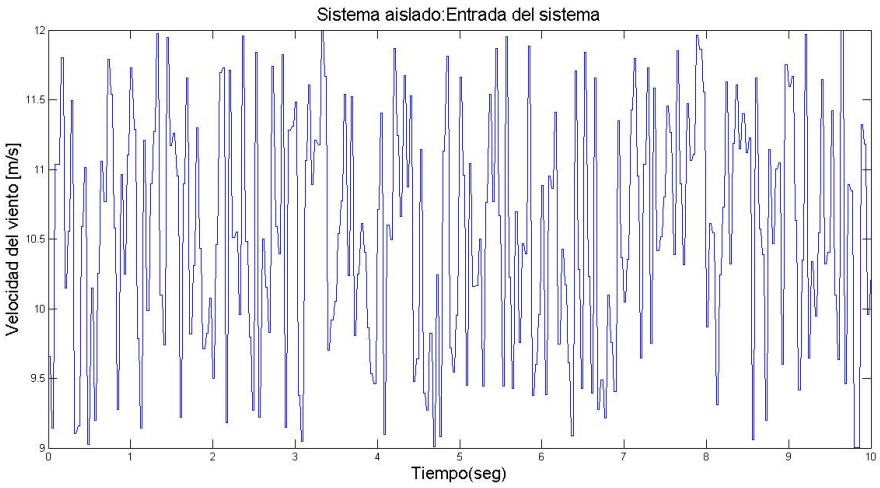

Figura 3.17: Entradas del sistema de generación aislado

Parque eólico

En esta sección se presenta la identificación de un aerogenerador de un parque eólico que exporta la potencia generada a una red de 120 KV a través de un sistema de distribución de 25 KV. El parque eólico consta de seis turbinas eólicas de 1.5 MVA. Las turbinas usan generadores de inducción jaula de ardilla, el estátor esta conectado directamente a la red y el rotores accionado por medio de una turbina eólica con un ángulo pitch variable. El angulo pitch es regulado a través de un controlador PI con el objetivo de limitar la potencia generada por la turbina para velocidades eólicas que excedan la velocidad nominal de 9 m/s. Cada turbina tiene un sistema de protección que monitorea el voltaje,la corriente y la velocidad de la máquina.

En identificación del aerogenerador se toman como señales de entrada la velocidad del viento y el ángulo pitch, y se toma como salida la potencia generada por el aerogenerador.

En la figura 3.21 se puede observar la señal de entrada del sistema, en la figura 3.22 se puede observar la identificación lineal del sistema usando el algoritmo CCA y en la figura 3.23 se puede observar la identificación del sistem,usando un modelo Hammerstein-wiener.

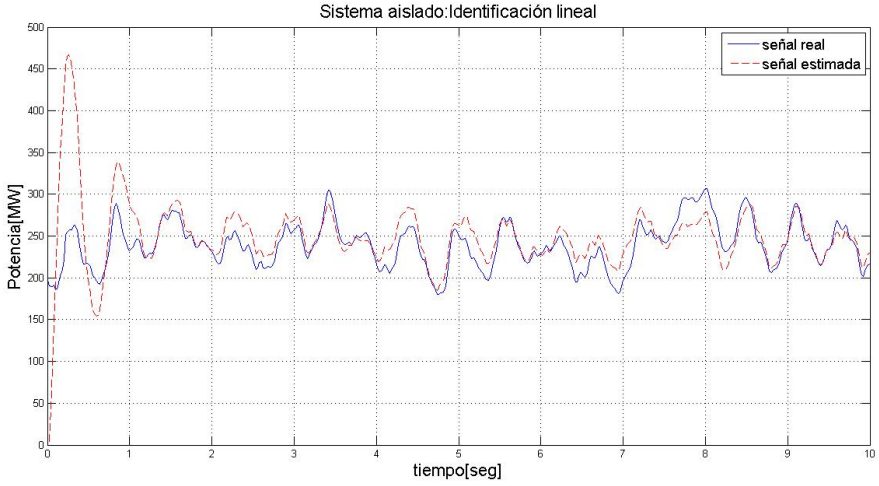

Figura 3.18: Identificación lineal del sistema de generación aislado

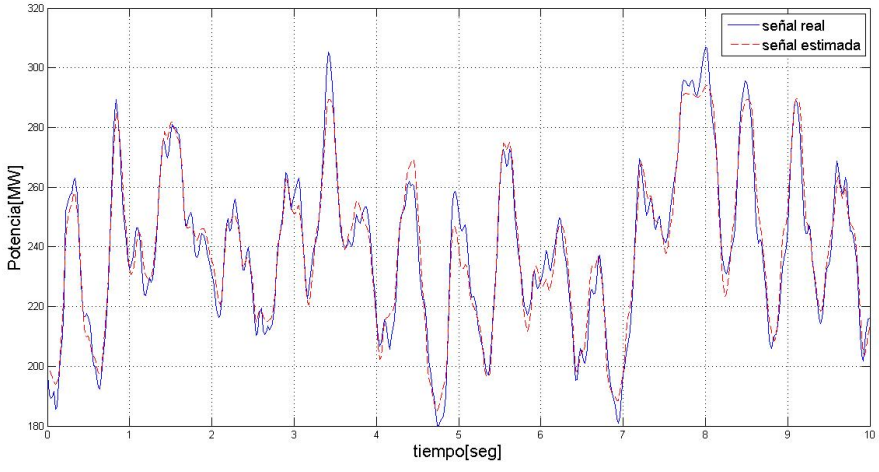

Figura 3.19: Identificación no lineal del sistema aislado usando un modelo Hammerstein-wiener

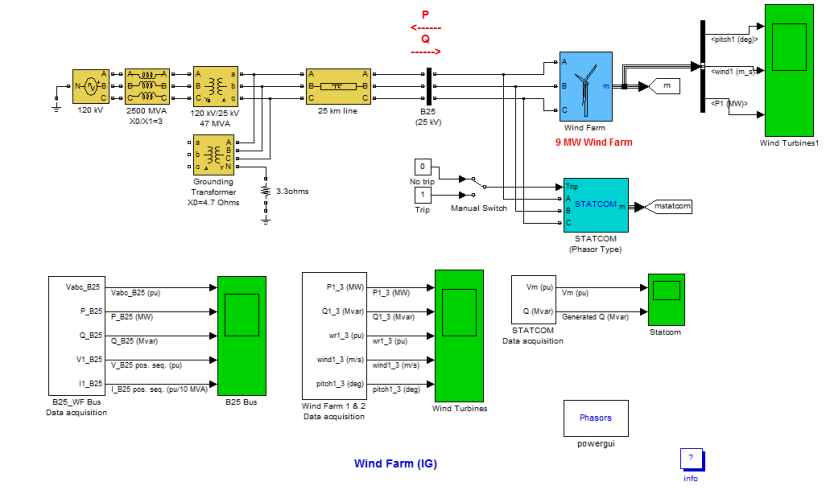

Figura 3.20: Sistema de generación eólico conectado al sistema de potencia

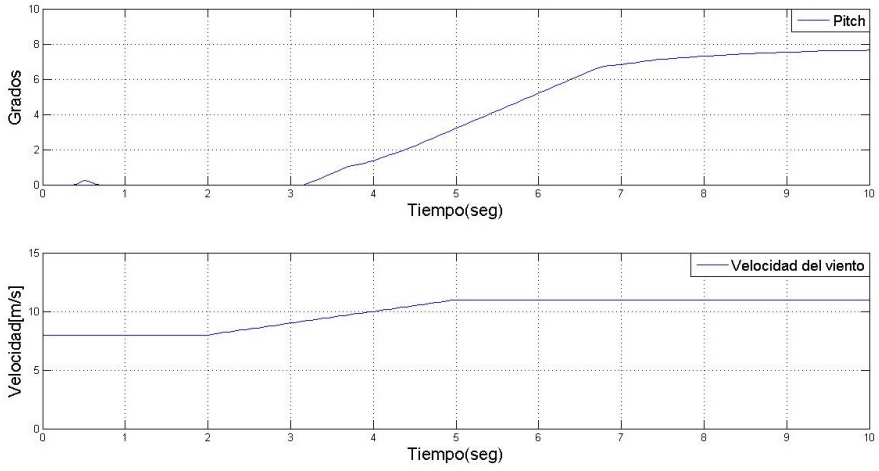

Figura 3.21: Entradas del sistema conectado al SP

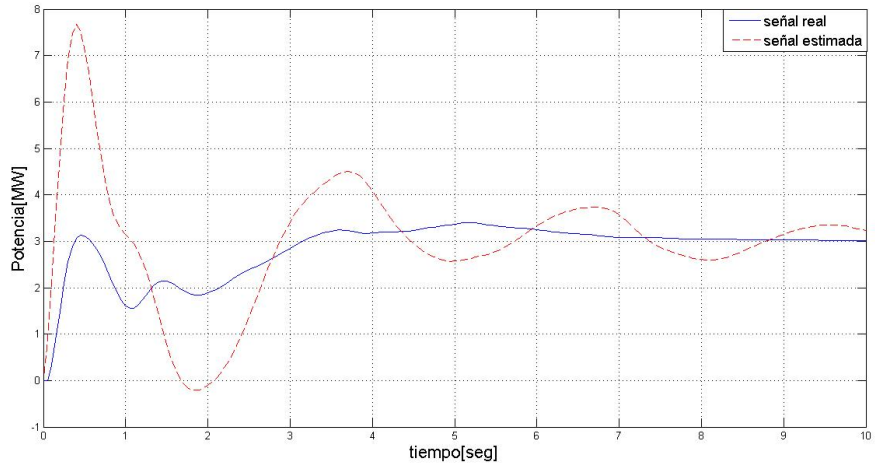

Figura 3.22: Identificación lineal del sistema conectado al SP

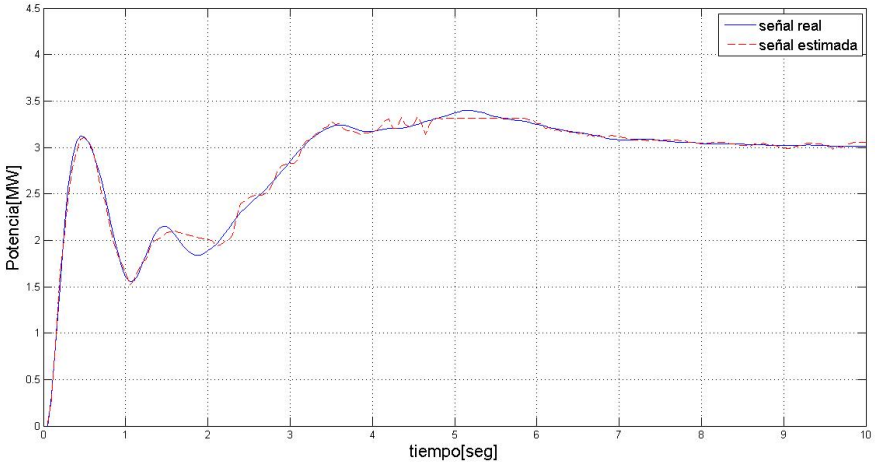

Figura 3.23: Identificación no lineal del sistema conectado al SP usando un modelo Hammerstein-Wiener

Capítulo 4

Algoritmos

4.1. Algoritmos de identificación de sistemas lineales

4.1.1. Bloque de Hankel

```
H = blkhank(y,i,j
function H = blkhank(y,i,j)

%Dimeniones del bloque
[l,nd] = size(y);
if nd < l;y = y';[l,nd] = size(y);end

% Check dimensions
if i < 0;error('blkHank: i debe ser positivo');end
if j < 0;error('blkHank: j debe ser positivo');end
if j > nd-i+1;error('blkHank: j demasiado grande');end

% Construcción del bloque de Hankel
H=zeros(l*i,j);
for k=1:i
    H((k-1)*l+1:k*l,:)=y(:,k:k+j-1);
end
```

4.1.2. Algoritmo n4sid usando descomposición RQ

```
%           [A,B,C,D] = n4sid(y,u,i);
%
%   Inputs:
%           y: salidas
%           u: entradas
%           i: filas delbloque de Hankel
%
%   Outputs:
%           A,B,C,D: Matrices del sistema en Espacio de estados
%
%                   x_{k+1) = A x_k + B u_k
%                     y_k   = C x_k + D u_k
%
```

```
%
%

%   Referencia:
%
%           Subspace Identification for Linear Systems
%           Peter Van Overschee / Bart De Moor
%           Kluwer Academic Publishers, 1996, Page 52
%


function [A,B,C,D] = n4sid(y,u,i,n);

if (nargin < 6);sil = 0;end



%Errores
[l,ny] = size(y);if (ny < l);y = y';[l,ny] = size(y);end
[m,nu] = size(u);if (nu < m);u = u';[m,nu] = size(u);end
if (i < 0);error('Número de filas del bloque de Hankel debe ser positivo');end
if (l < 0);error('vector de salidas vacio');end
if (m < 0);error('vector de entradas vacio');end
if (nu ~= ny);error('El número de datos de entrada y salidas es diferente');end
if ((nu-2*i+1) < (2*l*i));error('El número de datos es  insuficiente');end
% Número de columnas del bloque de Hankel matrices
j = nu-2*i+1;



% Descomposición RQ

  U = blkhank(u/sqrt(j),2*i,j);         %bloque de Hankel de entradas U
  Y = blkhank(y/sqrt(j),2*i,j);         % Bloque de Hankel de salidas
  R = triu(qr([U;Y]'))';        % factor R
  R = R(1:2*i*(m+l),1:2*i*(m+l));
  clear U Y


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%





% ************************************
%              PASO 1
% ************************************

mi2 = 2*m*i;
% Set up some matrices

  Rf = R((2*m+l)*i+1:2*(m+l)*i,:);   % salidas futuras
  Rp = [R(1:m*i,:);R(2*m*i+1:(2*m+l)*i,:)]; % salidas pasadas y futuras
  Ru  = R(m*i+1:2*m*i,1:mi2);        % entradas futuras
  % matriz perpendicular de las  salidas futuras
  Rfp = [Rf(:,1:mi2) - (Rf(:,1:mi2)/Ru)*Ru,Rf(:,mi2+1:2*(m+l)*i)];
  % Perpendicular Past
  Rpp = [Rp(:,1:mi2) - (Rp(:,1:mi2)/Ru)*Ru,Rp(:,mi2+1:2*(m+l)*i)];
```

```
% Proyección oblicua:

 Ob = (Rfp/Rpp)*Rp;



% ***************************************
%               PASO 2
% ***************************************

% Cálculo de la descompocisión en valores singulares
  [U,S,V] = svd(Ob);
  ss = diag(S);
  clear V S WOW



% ***************************************
%               PASO 3
% ***************************************

% Orden del sistema
  maximo=size(ss);
  acum=cumsum(ss);
  for k=1:maximo(1)
      n=k;
      if (acum(k)/acum(end))>0.9
          break
      end

  end


U1 = U(:,1:n);

clear  maximo acum


% ***************************************
%               PASO 4
% ***************************************

% Determine gamma
gam  = U1*diag(sqrt(ss(1:n)));
gamm = gam(1:l*(i-1),:);
% Calculamos pseudo-inversas
gam_inv  = pinv(gam);
gamm_inv = pinv(gamm);
clear gam gamm



% ***************************************
%               PASO 5
% ***************************************

% Calculamos segunda proyección oblicua Obm
Rf = R((2*m+l)*i+l+1:2*(m+l)*i,:);   % Future outputs
Rp = [R(1:m*(i+1),:);R(2*m*i+1:(2*m+l)*i+l,:)];
Ru  = R(m*i+m+1:2*m*i,1:mi2);        %
Rfp = [Rf(:,1:mi2) - (Rf(:,1:mi2)/Ru)*Ru,Rf(:,mi2+1:2*(m+l)*i)];
```

```
Rpp = [Rp(:,1:mi2) - (Rp(:,1:mi2)/Ru)*Ru,Rp(:,mi2+1:2*(m+l)*i)];

  Obm = (Rfp/Rpp)*Rp;


% Determinamos los estados
Xi  = gam_inv  * Ob;
Xip = gamm_inv * Obm;
clear gam_inv gamm_inv Obm


% ************************************
%               PASO 6
% ************************************


% Determinamos el sistema de matrices [A,B,C,D]

Rhs = [       Xi    ;  R(m*i+1:m*(i+1),:)]; % Lado derecho del sistema
Lhs = [      Xip    ;  R((2*m+l)*i+1:(2*m+l)*i+l,:)]; % Lado izquierdo del sistema

sol = Lhs/Rhs;

% Extraemos el sistema de matrices
A = sol(1:n,1:n);
B = sol(1:n,n+1:n+m);
C = sol(n+1:n+l,1:n);
D = sol(n+1:n+l,n+1:n+m);
```

4.1.3. Algortitmo N4SID usando teoría de mínimos cuadrados

```
function [A,B,C,D] = N4SID(y,u,i);


%Errores
[l,ny] = size(y);if (ny < l);y = y';[l,ny] = size(y);end
[m,nu] = size(u);if (nu < m);u = u';[m,nu] = size(u);end
if (i < 0);error('Número de filas del bloque de Hankel debe ser positivo');end
if (l < 0);error('vector de salidas vacio');end
if (m < 0);error('vector de entradas vacio');end
if (nu ~= ny);error('El número de datos de entrada y salidas es diferente');end
if ((nu-2*i+1) < (2*l*i));error('El número de datos es  insuficiente');end
% Número de columnas del bloque de Hankel matrices
j = nu-2*i+1;


%Cálculamos bloques de Hankel

  U = blkhank(u,2*i,j);
  Y = blkhank(y,2*i,j);
  Yp=Y(1:i*l,:);
  Yf=Y(i*l+1:2*i*l,:);
  Up=U(1:i*m,:);
  Uf=U(i*m+1:2*i*m,:);

  Ypt=Y(1:(i+1)*l,:);
  Yf_=Y((i+1)*l+1:2*i*l,:);
  Upt=U(1:(i+1)*m,:);
  Uf_=U((i+1)*m+1:2*i*m,:);

%Cálculamos proyecciones oblicuas usando minimos cuadrados
%  x=[Up;Uf;Yp];
%  b=Yf;
 [LuLy]=(inv(([Up;Uf;Yp]*[Up;Uf;Yp]'))*[Up;Uf;Yp]*Yf')';
```

```
 Lu=LuLy(:,1:2*i*m);
 Ly=LuLy(:,2*i*m+1:2*i*m+i*l);
 size(Lu)
 size(Ly)
 Ob=Lu(:,1:m*i)*Up+Ly*Yp;

 [Lu_Ly_]=(inv(([Upt;Uf_;Ypt]*[Upt;Uf_;Ypt]'))*[Upt;Uf_;Ypt]*Yf_')';
 Lu_=Lu_Ly_(:,1:2*i*m);
 Ly_=Lu_Ly_(:,2*i*m+1:2*i*m+(i+1)*l);
 Obm=Lu_(:,1:(i+1)*m)*Upt+Ly_*Ypt;


 % ************************************
%              PASO 2
% ************************************

% Cálculamos  SVD de la proyección oblicua

  [Us,S,V] = svd(Ob);
  ss = diag(S);
  clear V S WOW




% ************************************
%              PASO 3
% ************************************

% Determinamos el oreden del sistema a partir de los valores singulares
  maximo=size(ss);
  acum=cumsum(ss);
  for k=1:maximo(1)
      n=k;
      if (acum(k)/acum(end))>0.9
          break
      end

  end



U1 = Us(:,1:n);                 % Determine U1

clear  maximo acum


% ************************************
%           Determinamos Gammas
% ************************************


gam  = U1*diag(sqrt(ss(1:n)));
gamm = gam(1:l*(i-1),:);
% Determinamos pseudo-inversas de Gammas
gam_inv  = pinv(gam);
gamm_inv = pinv(gamm);
clear gam gamm ss



% ************************************
%       Cálculo de los Estados
% ************************************
```

```
Xi  = gam_inv  * Ob;
Xip = gamm_inv * Obm;
clear gam_inv gamm_inv Obm


% ************************************
%   Cálculo del sistema de matrices
% ************************************



Rhs = [       Xi   ;  U(m*i+1:m*(i+1),:)]; %Parte derecha del sistema de ecuaciones
Lhs = [      Xip   ;  Y(l*i+1:l*(i+1),:)]; % Parte izquierda del sistema de ecuaciones

sol = Lhs/Rhs;

% Extracción del sistema de matrices
A = sol(1:n,1:n);
B = sol(1:n,n+1:n+m);
C = sol(n+1:n+l,1:n);
D = sol(n+1:n+l,n+1:n+m);
```

4.2. Algoritmos de identificación de sistemas no lineales

4.2.1. Algoritmo de identificacion de un modelo Hammerstein

```
%%algoritmo identificación de un sistema miso usando un modelo hammerstein%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



num_11=[1,0.8,0.3,0.4,0,0,0];
num_12=[1,0.9,0.7,0.2,0,0,0];
den_11=poly([0.98*exp(i),0.98*exp(-i),0.98*exp(1.6*i),0.98*exp(-1.6*i),0.97*exp(0.4i),0.97*exp(-0.4*i)]);
den_12=den_11;

nums={num_11,num_12}
dens={den_11,den_12}
sys=tf(nums,dens,1);

%Entrada u:white gaussian noise de tamaño(L=400),desviacion estandar(sigma=2) y media(mu=0)
L=400; %Sample length for the random signal
sigma=2;
u=sigma*randn(2,L);




%
for k=1:400
f(1,k)=sinc(u(1,k))*(u(2,k)^2);
f(2,k)=sinc(u(1,k))-sinc(u(2,k));
end
y=lsim(sys,f);

%*************************************************************************



i= 10;
[l,ny] = size(y);if (ny < l);y = y';[l,ny] = size(y);end
```

```
[m,nu] = size(u);if (nu < m);u = u';[m,nu] = size(u);end
if (nu ~= ny);error('Number of data points different in input and output');end
if ((nu-2*i+1) < (2*l*i));error('Not enough data points');end
N=ny;
% Número de columnas de los bloques de Hankel
j = nu-2*i+1;

 U = blkhank(u/sqrt(j),2*i,j);
 Y = blkhank(y/sqrt(j),2*i,j);
Yp=Y(1:i*l,:);
Yf=Y(i*l+1:2*i*l,:);
Ypt=Y(1:(i+1)*l,:);
Yf_=Y((i+1)*l+1:2*i*l,:);
Up=U(1:i*m,:);
Uf=U(i*m+1:2*i*m,:);

%%%%%%%%%%%%%%%%%%%%calculo Kp y Kf%%%%%%%%%%%%%%%%%%%%%%%%

sigma=0.9; %********************Kernel Gaussiano**********************
gamma=500; %********************GAMMA**********************
Gbd=5;     %********************GAMMA_{BD}**********************

%%%%%%%%%%%%%%%%%%%%%%%%   Kp     %%%%%%%%%%%%%%%%%%%%%%%%

Kp=zeros(j,j);
for p=1:j
    for q=1:j
        for h=1:i
            Kp(p,q)=Kp(p,q)+exp(-norm( (u(:,h+p-1) - u(:,h+q-1) ))^2/(2*sigma^2));
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%   Kp^{+}     %%%%%%%%%%%%%%%%%%%%%%%%
Kpt=zeros(j,j);
for p=1:j
    for q=1:j
        for h=1:i+1
            Kpt(p,q)=Kpt(p,q)+exp(-norm( u(:,h+p-1) - u(:,h+q-1))^2/(2*sigma^2));
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%   Kf     %%%%%%%%%%%%%%%%%%%%%%%%
Kf=zeros(j,j);

for p=1:j
    for q=1:j
        for h=i+1:2*i
            Kf(p,q)=Kf(p,q)+exp(-norm( (u(:,h+p-1) - u(:,h+q-1)))^2/(2*sigma^2));
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%OMEGA     %%%%%%%%%%%%%%%%%%%%%%%%
for p=1:N
    for q=1:N
        OMEGA(p,q)=exp((-norm( u(:,p) - u(:,q))^2)/(2*sigma^2));
    end
end



%%%%%%%%%%%%%%%%%%%%SISTEMA LEMMA 3.1 %%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
PI=[zeros(i*l,i*l),Yp;Yp',Kp+Kf+(1/gamma)*eye(j)];
PD=[zeros(i*l,i*l);Yf'];
LyA=inv(PI)*PD;




Ly=LyA(1:i*l,:)';
Ac=LyA(i*l+1:end,:); %A cursiva que contiene multiplicadores de lagrange alpha
Oi=Ac'*Kp+Ly*Yp;
Ob=Oi;



%CALCULO PROYECCIÓN OBLICUA O_{i+1}

PI=[zeros((i+1)*l,(i+1)*l),Ypt;Ypt',Kp+Kf+eye(j)];
PD=[zeros((i+1)*l,(i—1)*l);;Yf_'];
LyA_=inv(PI)*PD;



 Ly_=LyA_(1:(i+1)*l,:)';
 Ac_=LyA_(1+(i+1)*l:end,:); %A cursiva que contiene multiplicadores de lagrange alpha

Oit1=(Ac_)'*Kpt'+Ly_*Ypt;
Obm=Oit1;


% CALCULO DE LAS SECUENCIAS DE ESTADO X_{i} Y X_{i+1}:
% Compute the SVD

  [Us,Ss,V] = svd(Ob);
  ss = diag(Ss);
  clear V Ss WOW




% ***********************************************
% Calculo del orden usando los valores singulares
% ***********************************************


  maximo=size(ss);
  acum=cumsum(ss);
  for k=1:maximo(1)
      n=k;
      if (acum(k)/acum(end))>0.95
          break
      end

  end



U1 = Us(:,1:n);

clear  maximo acum


% ************************************
%        Determinamos Gammas
% ************************************
```

```
gam  = U1*diag(sqrt(ss(1:n)));
gamm = gam(1:l*(i-1),:);
% And their pseudo inverses
gam_inv  = pinv(gam);
gamm_inv = pinv(gamm);
clear gam gamm ss



% ************************************
% Determinamos los estados del sistema
% ************************************


Xi  = gam_inv  * Ob;
Xip = gamm_inv * Obm;
clear gam_inv gamm_inv Obm

%*****************************************
%      Extracción sistema de matrices
%*****************************************



%*****************************************
%******************Kbd********************

for p=1:j
    for q=1:j

            Kbd(p,q)=exp((-norm( u(:,i+p-1) - u(:,i+q-1) )^2)/(2*sigma^2));

    end
end


%********CALCULO SISTEMA DE MATRICES******************

PI=[zeros(n,n),Xi;Xi',Kbd+(1/Gbd)*eye(j)];
PD=[zeros(n,n+l);[Xip;Y(i*l+1:(i+1)*l,:)]'];

AC_Abd=inv(PI)*PD;
Abd=AC_Abd(n+1:end,:);


AC=AC_Abd(1:n,:)';


BD_f=Abd'*OMEGA(i+1:i+j,:);

BD=(pinv(f')*BD_f')';



An=AC(1:n,:);
Cn=AC(n+1:n+l,:);
Bn=BD(1:n,:);
Dn=BD(n+1:n+l,:);

yn=lsim(ss(An,Bn,Cn,Dn,1),f);


figure(1)
```

```
plot(t,y,'b',t,yn,'--r')
title('Identificación no lineal','FontSize', 20)
leyenda=legend('salida generada','salida estimada');
set(leyenda,'FontSize', 14)
xlabel('Muestras','FontName','Arial','FontSize', 16)
ylabel('Amplitud','FontName','Arial','FontSize', 16)



figure(2)
subplot(211)
plot(t,u)
title('Señales de entrada ','FontSize', 20)
xlabel('Muestras','FontName','Arial','FontSize', 16)
ylabel('Amplitud','FontName','Arial','FontSize', 16)
leyenda=legend('Entrada 1')
set(leyenda,'FontSize', 14)

subplot(212)
plot(t,u(2,105:204))
xlabel('Muestras','FontName','Arial','FontSize', 16)
ylabel('Amplitud','FontName','Arial','FontSize', 16)
leyenda=legend('Entrada 2')
set(leyenda,'FontSize', 14)
```

Conclusiones

- Los algoritmos N4SID y CCA son sistemas de identificación con un fuerte soporte matemático, basado en conceptos geométricos, matriciales y algebraicos que eliminan problemas de óptimos locales y procesos iterativos, reduciendo significativamente el costo computacional.
- Los algoritmos N4SID y CCA permiten la identificación de sistemas lineales MIMO con un alto desempeño, a partir únicamente de las señales de entrada y salida, evitando el modelado de los sistemas y conocimiento del orden del sistema, necesario en la mayoría de los métodos de identificación.
- En la identificación de sistemas no lineales, se pudo observar que los algoritmos N4SID y CCA siendo necesario en este tipo de sistemas algoritmos de identificación no lineal.
- La extensión del algoritmo N4SID para la identificación de sietemas no lineales a través de la teoria de máquinas de soporte vectorial LS-SVM presenta un alto desempeño y reduce significativamente el costo computacional de la identificación debido a que no utiliza ningún proceso iterativo, sin embargo en este algoritmo es necesario conocer a priori la función no lineal del bloque estático de las señalas de entrada f.
- La identificación no lineal utilizando modelos Hammerstein-Wiener a través de procesos iterativo permiten la identificación con un gran desempeño, sin embargo, debido a los procesos iterativos utilizados en la identificación el costo computacional es mayor que en el algoritmo N4SID extendido a sistemas no lineales.
- La identificación de sistemas de generación eólica a través de los algoritmos de identificación lineal es insuficiente debido a que presentan un desempeño muy bajo.
- La identificación de sistemas de generación eólica a través de los algoritmos de identificación lineal usando el algoritmo CCA es insuficiente debido a que presentan un desempeño muy bajo.
- La identificación de sistemas de generación eólica a través de modelos Hammerstein-Wiener presentan un alto desempeño comparado con la identificación lineal, sin embargo su costo computacional es mas alto.

Bibliografía

[1] E. Giraldo, D. Giraldo; Teoría de control análogo; Universidad Tecnológica de Pereira, 2010.

[2] Subspace Identification of Hammerstein-Wiener systems using Kernel Canonical Correlation Analysis.

[3] Van Overschee P., De Moor B., *Unifying theorem for three Subspace System Identification Algorithms*. Automatica, Special issue on trends in system identification. vol.31, No 12, 1995.

[4] M. Vijayalaxmi, N. Shanmuga Vadivoo; Identification of Doubly Fed Induction Generator based Wind Energy Conversion System Using Piecewise-Linear Hammerstein Wiener Model. Department of Electrical and Electronics Engineering, Thiagarajar College of Engineering, Madurai, India.